AD-A060 228 NAVY UNDERWATER SOUND LAB NEW LONDON CONN F/G 9/2
A GENERALIZED REAL-TIME EXECUTIVE ROUTINE FOR THE UNIVAC 1230 C--ETC(U)
MAR 70 D POTTER, G BOTSEAS, C BECKER

UNCLASSIFIED NUSL-TM-2211-63-70 NL

| OF |
AD
A060228

END
DATE
FILMED
12-78
DDC

NUSL Problem No.
0-A-408-00-00
0-A-401-00-00

NAVY UNDERWATER SOUND LABORATORY
NEW LONDON, CONNECTICUT

(6) A GENERALIZED REAL-TIME EXECUTIVE ROUTINE FOR THE UNIVAC 1230 COMPUTER.

by

(10) David Potter, George Botseas and Clair Becker

NUSL Technical Memorandum No. 2211-63-70

(11) 31 March 1970

INTRODUCTION

Use of a computer to perform repetitive real-time tasks such as equipment control, monitoring, data gathering and data reduction requires an executive routine which will ensure that the proper actions are initiated at the appropriate times.

This memorandum describes GPEXEC1, a generalized real-time executive routine for the Univac 1230 computer. Written in assembly language, it can be assembled by and run on other Univac computers which are similar to the 1230.

Included is a flow chart, a sample timing chart, a program listing and a timing test routine.

GENERAL DESCRIPTION

(12) 10 p.

GPEXEC1 is a program which calls on a sequence of routines in a predetermined order. The time at which each routine in the sequence is called is determined by preset table entries and the computer's real-time clock.

THE REAL-TIME CLOCK

The real-time clock is, for programming purposes, a memory cell whose contents are increased by one every 1024th of a second. Since the clock will overflow into the sign bit and turn negative after approximately six days, it is necessary to reset it periodically. Therefore, a check on the clock is made prior to the execution of each

routine. If a reset is necessary, a flag is set and the clock is cleared following the execution of the upcoming routine.

CAPABILITIES OF GPEXEC1

GPEXEC1 is capable of repeatedly executing a sequence of N routines in a specified order. The time of execution for each routine, relative to the start time of the exec, is specified by the user. Also specified by the user is REPTIME, the time interval between successive executions of the sequence of routines.

It is assumed that each routine will be completed before it is time to execute the next. However, should a routine fail to return control to the exec prior to the execution time of the next, the next routine will be executed immediately upon return of control. The manner in which the timing is handled will ignore this tardiness and the routine will be scheduled for its next execution at the proper time.

The time of execution of any routine may be varied relative to the preceding and following routines. Also, the entire sequence may be easily shifted forward or backward in time.

GPEXEC1 can be set up so that one or more routines are executed only occasionally. A routine can also be deleted entirely from the sequence of events. These capabilities will be discussed in detail later in this memorandum.

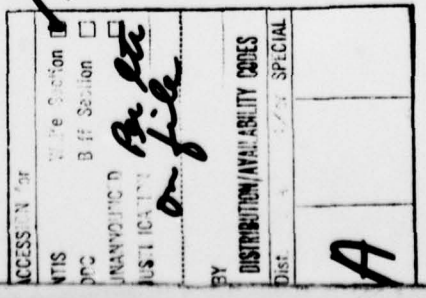
SETTING UP GPEXEC1 FOR A SPECIFIC TASK

To set up GPEXEC1 for a specific task, it is necessary to supply entries to two tables and to set two parameters.

The first parameter is the number punched on the card labeled EXEC 05 (see program listing). N should be set to the number of routines which the exec will control.

The second parameter is the number punched on the REPTIME card, labeled EXEC 54. REPTIME should be set to the number of clock cycles from the start of one sequence to the next.

The first table to be filled is the INITLTIME table, which must contain N entries. The jth entry is the number of clock cycles after the start of the exec that the jth routine is to be executed.



The other required table is called JUMPTABLE. It also must contain N entries, each of which must be a return jump (RJP) to the desired routine. Each entry in the JUMPTABLE may be coded with a keyset condition.

THEORY OF OPERATION

Once initiated, the exec transfers the contents of INITLTIME to a working area, TIMETABLE, and waits for a command to start. When this is received, the real-time clock is zeroed and a comparison is made between the clock and the first entry in TIMETABLE. When the clock reaches or exceeds this value, the table value is increased by REPTIME and the new value is checked for overflow. The corresponding routine in JUMPTABLE is executed, provided that the keyset condition is met or the entry has not been cleared.

If the overflow test indicated that a reset of the clock was necessary, the clock reading is subtracted from all entries in TIMETABLE and the clock is zeroed.

A check is then made to determine whether the program should be terminated. If so, the exec exits. If not, the routine index is incremented by one and the exec awaits the execution time of the next routine. Following the execution of the last routine in the sequence, the routine index is cleared so that the exec now waits for the proper time to re-execute the first routine in the sequence.

METHODS OF DELETING A ROUTINE

There are two ways in which a routine can be deleted. The most flexible is by including a keyset condition in the coding of the JUMPTABLE. Thus the execution would be deleted if the keyset condition were not met.

Another method is to simply zero out the desired cell in JUMPTABLE. The exec performs a test on the entries prior to any attempted execution and aborts if a zero is found.

CHANGING THE TIMING

The execution time of any or all routines may be advanced or retarded by increasing or decreasing the contents of the associated TIMETABLE cell(s) by the desired number of clock cycles.

GENERAL COMMENTS

The enclosed program listing is coded so that the exec may be started and terminated by either a key setting or the setting of a memory cell. The memory cell flag seems to be preferable since it can be set by interrupt from a keyboard or an external piece of equipment.

David Potter
DAVID POTTER
Mathematician

George Botseas
GEORGE BOTSEAS
Computer Specialist

Clair Becker
CLAIR BECKER
Mathematician

APPENDIX A

PROGRAM LISTING

```
TEST  01              C-CONTROL
TEST  02              ALLOCATION
TEST  03  BASE        10000
TEST  04  ENTRANCE    GPEXEC1
TEST  05              SYS-PROC*POTTER*FEB70
TEST  06              LOC-DD
TEST  07              VRBL*TIME*FXW*10D
TEST  08              END-LOC-DD
TEST  09              PROCEDURE*R1
TEST  10              PUT*W(160)*W(TIME)
TEST  11              TYPET*$CR$ROUTINE '
TEST  12              ENT*A*1+B1
TEST  13              TYPE-DEC*A*$SP$*$SP$'
TEST  14              TYPE$*TIME*THEN*RETURN
TEST  15              END-PROC*R1
TEST  16              PROCEDURE*R2
TEST  17              R1
TEST  18              RETURN
TEST  19              END-PROC*R2
TEST  20              PROCEDURE*R3
TEST  21              R1
TEST  22              RETURN
TEST  23              END-PROC*R3
TEST  24              PROCEDURE*R4
TEST  25              R1
TEST  26              TYPET*$CR$END OF CYCLE$CR$'
TEST  27              RETURN
TEST  28              END-PROC*R4
EXEC  01  GPEXEC1     PROGRAM*USN/USL
EXEC  02              COMMENT*GENERAL PURPOSE REAL-TIME EXECUTIVE
EXEC  03              COMMENT*ROUTINE FOR UNIVAC 1230 COMPUTER
EXEC  04  CLOCK       EQUALS*160
EXEC  05  N           EQUALS*4'CHANGE FOR NUMBER OF ROUTINES
EXEC  06              COMMENT*N IS THE NUMBER OF ROUTINES TO BE
EXEC  07              COMMENT*CONTROLLED BY THE EXECUTIVE
EXEC  08  GPEXEC1     ENTRY
EXEC  09              STR*B0*W(STARTEND)'INITIALIZE CONTROL FLAG
EXEC  10              ENT*B7*N-1'INITIALIZE
EXEC  11  EXEC1       ENT*A*W(INITLTIME+B7)'TIMETABLE
EXEC  12              STR*A*W(TIMETABLE+B7)
EXEC  13              BDP*B7*EXEC1
EXEC  14              COMMENT*INSERT ADDITIONAL INITIALIZING
EXEC  15              COMMENT*ROUTINES HERE
EXEC  16  EXEC2       ENT*A*0(STARTEND)*ANOT'TIME TO START
EXEC  17              JP*EXEC2*KEY1'NO  WAIT
EXEC  18              STR*B0*W(CLOCK)'YES  CLEAR CLOCK
EXEC  19              CL*B1'SUBROUTINE INDEX
EXEC  20  EXEC3       ENT*0*W(TIMETABLE+B1)'EXECUTION TIME
EXEC  21  EXEC4       ENT*Y-0*W(CLOCK)*APOS'TIME TO EXECUTE
EXEC  22              JP*EXEC4'NO  WAIT
EXEC  23              ENT*0*W(REPTIME)'YES
EXEC  24              RPL*Y+0*W(TIMETABLE+B1)'UPDATE TABLE
EXEC  25              STR*B0*W(CLOCKFLAG)'CLEAR RESET FLAG
EXEC  26              LSH*A*1*APOS'TEST FOR CLOCK RESET
EXEC  27              STR*B0*CPW(CLOCKFLAG)'RESET NEEDED
EXEC  28              STR*B1*L(EXEC0)'SAVE INDEX
EXEC  29              ENT*A*W(JUMPTABLE+B1)
EXEC  30              STR*A*W(EXEC3)*AZERO'ROUTINE DELETED
```

```
EXEC   31  EXEC5      0'NO  EXECUTE IT
EXEC   32  EXEC6      ENT*B1*0'RESTORE INDEX
EXEC   33             ENT*A*W(CLOCKFLAG)'RESET CLOCK
EXEC   34             JP*EXEC7*AZERO'NO
EXEC   35             ENT*G*W(CLOCK)'YES
EXEC   36             RPT*N*ADV'ADJUST
EXEC   37             RPL*Y-G*W(TIMETABLE)'TABLE
EXEC   38             STR*BO*W(CLOCK)'CLEAR CLOCK
EXEC   39  EXEC7      BSK*BO*L(STARTEND)'TERMINATE PROGRAM
EXEC   40             EXIT'YES
EXEC   41             EXIT*KEY2'YES  IF KEY2 SET
EXEC   42             BSK*B1*N-1'NO INCREMENT INDEX
EXEC   43             NO-OP
EXEC   44             JP*EXEC3
EXEC   45  INITLTIME  2048D'T1 - 2 SECONDS
EXEC   46             12800D'T2 - 12.5 SECONDS
EXEC   47             25856D'T3 - 25.25 SECONDS
EXEC   48             40832D'T4 - 39.875 SECONDS
EXEC   49  TIMETABLE  RESERVE*N'NUMBER OF ROUTINES
EXEC   50  JUMPTABLE  RJP*R1
EXEC   51             RJP*R2*KEY3
EXEC   52             RJP*R3
EXEC   53             RJP*R4
EXEC   54  REPTIME    102400D'TIME BETWEEN RECYCLES  100 SECONDS HERE
EXEC   55  CLOCKFLAG  0'CLOCK RESET FLAG
EXEC   56  STARTEND   0'UPPER START,LOWER END
                      END-DATA
```

APPENDIX B

TIMING AND FLOW CHARTS

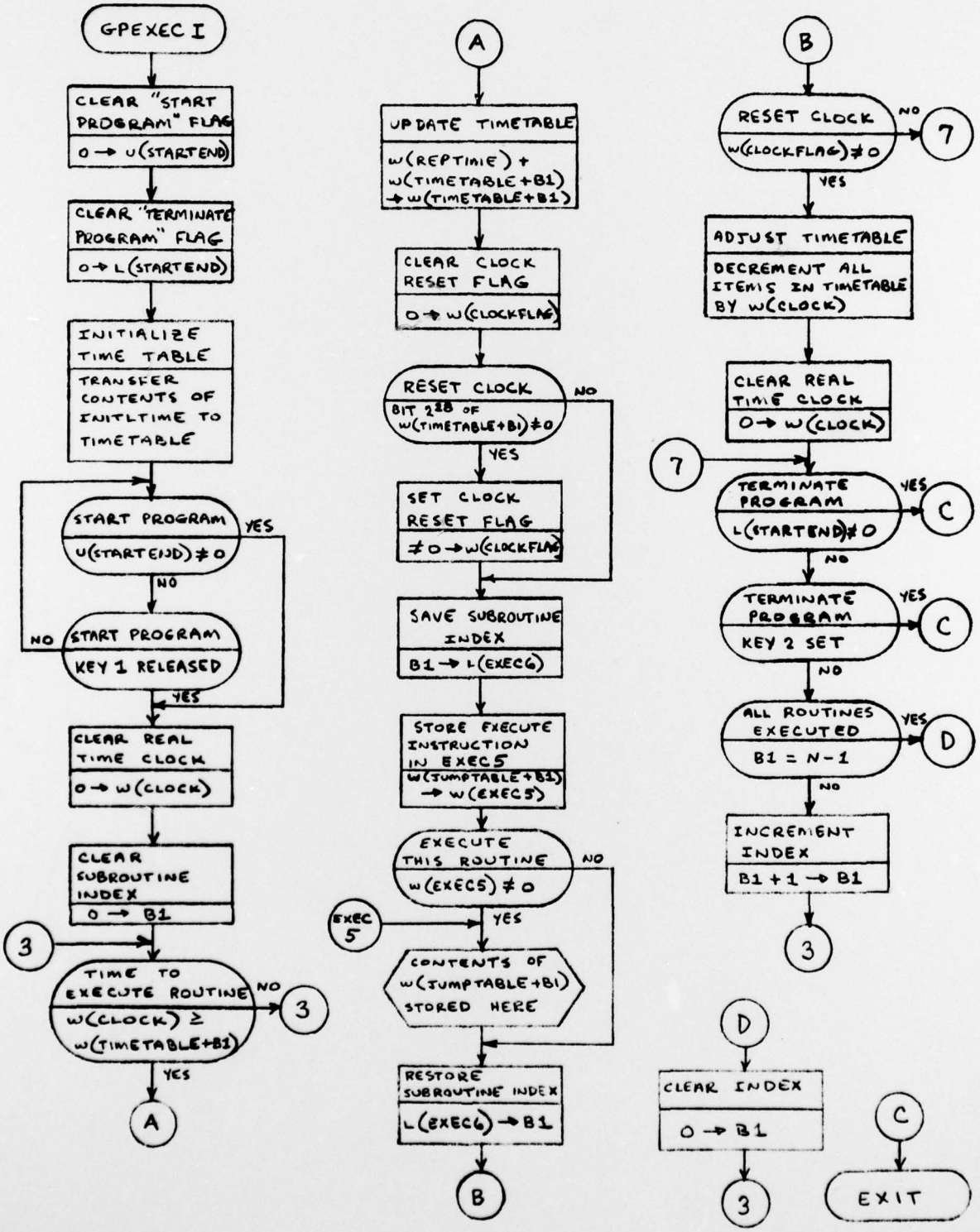
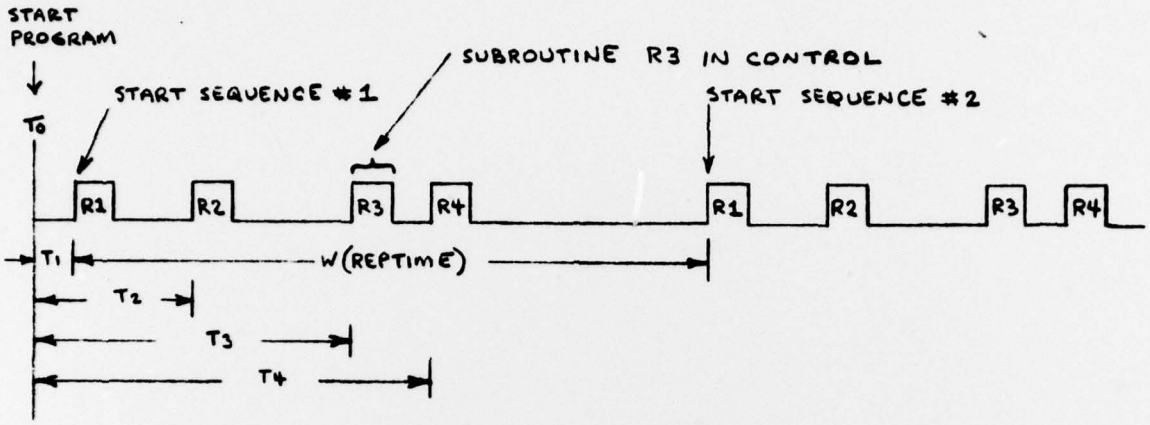


FIGURE 1 - DETAILED FLOWCHART OF GPEXECI



INITLTIME
T_1
T_2
T_3
T_4

TIMETABLE
T_1
T_2
T_3
T_4

BEFORE SEQUENCE #1

TIMETABLE
T_1+w(REPTIME)
T_2+w(REPTIME)
T_3+w(REPTIME)
T_4+w(REPTIME)

AFTER SEQUENCE #1

JUMPTABLE
RJP * R1
RJP * R2 * KEY3
RJP * R3
RJP * R4

NOTES

1. THE ENTIRE SEQUENCE OF SUBROUTINES MAY BE SHIFTED EARLIER/LATER IN TIME BY DECREMENTING/INCREMENTING ALL ITEMS IN TIMETABLE BY Δt.

2. INDIVIDUAL SUBROUTINES MAY BE SHIFTED EARLIER/LATER IN TIME BY DECREMENTING/INCREMENTING THE CORRESPONDING ITEM IN TIMETABLE BY Δt.

3. SUBROUTINE R2 WILL BE EXECUTED ONLY IF KEY 3 IS SET.

4. INDIVIDUAL SUBROUTINES MAY BE DELETED BY STORING A ZERO IN THE CORRESPONDING ITEM IN JUMPTABLE.

FIGURE 2 - SAMPLE TIMING CHART AND CONTENTS OF EXEC TABLES